VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Technologie WinRT

WinRT Technology

2013

Jakub Dvořák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Dvořák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Technologie WinRT
WinRT Technology

Zásady pro vypracování:

Cílem práce je zmapovat a popsat platformu WinRT (Windows 8) a následně implementovat ukázkovou aplikaci pro toto prostředí využívající jeho specifika.

1. Zanalyzujte a popište platformu WinRT z pohledu architektury a technologií.
2. Popište platformu z pohledu vývoje.
3. Navrhněte ukázkovou aplikaci, která bude hojně využívat specifické vlastnosti platformy. Bude se jednat o aplikaci využívající plně možnosti uživatelského rozhraní a zároveň využívající datovou konektivitu pro přístup k datům, která bude vizualizovat.
4. Implementujte tuto aplikaci.
5. Zhodnoťte možnosti převodu aplikace z platformy Windows Phone do prostředí WinRT. Toto ilustrujte na konkrétní aplikaci.
6. Zhodnoťte vlastnosti a přínosy platformy WinRT.

Seznam doporučené odborné literatury:

[1] M.B.Raynolds: An Introduction to Windows Runtime (WinRT), @mbrit, 2012
[2] A.Freeman: Metro Revealed: Building Windows 8 apps with XAML and C#, ISBN: 978-1430244912, Apress, 2012

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2013 .....

Rád bych touto cestou poděkoval Ing. Michalovi Radeckému, Ph.D. za cenné rady a odborné a trpělivé vedení mé bakalářské práce.

Abstrakt

Počítače a čím dál tím více také mobilní zařízení se stávají naší neoddělitelnou součástí. Tomuto faktu nasvědčuje i zvyšující se statistika počtu osobních počítačů a mobilních zařízení na jednu osobu. Konkrétně užití mobilních zařízení penetruje dnešní trh a oblasti průmyslu jako je vzdělávání, medicína, obchod a další mnohem více než kdykoliv v minulosti. Tato bakalářská práce má za cíl představit technologii Windows Runtime společnosti Microsoft z pohledu vývoje Windows Store aplikací a toto ilustrovat na ukázkové aplikaci. Na této aplikaci jsou ukázány důležité součásti aplikace, specifika designových prvků a rozdíly vývoje oproti Win32 aplikacím. Práce také obsahuje možnosti vylepšení aplikace pro další iterace.

Klíčová slova: Windows Runtime, Windows 8, Windows Store aplikace, C#, XAML, bakalářská práce

Abstract

Personal computers and even more mobile devices are getting our inseparable daily experience. This fact also suggests the increasing statistics of personal computers and mobile devices per head. Concretely usage of mobile devices penetrates today s market and industry such as education, healthcare ans retail much more often than ever before. The goal of this bachelor thesis is to present Microsofts Windows Runtime technology in terms of developing Windows Store apps and demonstrate that on the example application. There are shown important components of this application, specifics of design layout and differences in contrast to Win32 applications. The paper also includes possible improvements for next iterations.

Keywords: Windows Runtime, Windows 8, Windows Store app, C#, XAML, bachelor thesis

Seznam použitých zkratek a symbolů

A-GPS	–	Assisted Global Positioning System
API	–	Application Programming Interface
BTS	–	Base Transceiver Station
CPU	–	Central processing unit
CSS	–	Cascading Style Sheets
DPI	–	Dots per inch
GPU	–	Graphics processing unit
HLSL	–	High Level Shader Language
HTML	–	Hyper Text Markup Language
IDE	–	Integrated Development Environment
PC	–	Personal Computer
PDF	–	Portable Document Format
RAM	–	Random-access memory
SVG	–	Scalable Vector Graphics
UI	–	User Interface
WinRT	–	Windows Runtime
WP	–	Windows Phone
WPF	–	Windows Presentation Found
XAML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Problematika vývoje aplikací pro osobní počítače a mobilní zařízení je v současné době velice obsahlou a probíranou kapitolou. Svět i vytváření aplikací se od prvních programů na děrných štítcích v mnohém změnil. Novodobé počítace a mobilní zařízení v podobě tzv. chytrých telefonů či tabletů hrají velice důležitou roli v každodenním životě téměř každého z nás. Jsou nedílnou součástí nejen pracovních povinností a vzájemné komunikace, ale poskytují nám také centrum zábavy a multimédií.

Současným trendem je minimalizace velikosti zařízení za ponechání stejného či podobného výkonu. Uživatelé stále častěji touží řešit své povinnosti a bavit se nejen doma, či v práci u stolního počítače, ale odkudkoliv je napadne. Toto je výzvou jak pro samotné výrobce hardwaru, tak i pro vývojáře operačních systémů a aplikací pro tato zařízení.

Existuje celá řada platforem, pro které může dnešní vývojář aplikace vytvářet. Jednou z nich je právě platforma Windows od společnosti Microsoft, která má dominantní podíl na trhu s operačními systémy. Uplynulo už více než dvacet let od prvního OS z dílny Microsoftu a aktuálně poslední oficiální verzí je Windows 8 či odlehčená varianta Windows RT. Budoucnost operačních systémů Microsoftu leží ve Windows 8 a jeho budoucím nástupci s kódovým označením Blue. Cílem této bakalářské práce je popsat vývoj aplikací pro Windows 8 pomocí technologie Windows Runtime a demonstrovat jej na ukázkovém příkladě.

V první kapitole představuji technologii Windows Runtime, kde je zmíněna základní charakteristika a jaké jsou prerekvizity pro vývoj aplikací. V následujících dvou kapitolách popisuji vlastnosti a vývoj samotných aplikací pro Windows 8. Po ní ukazuji možnosti migrace aplikace z Windows Phone do Windows 8 na konkrétní aplikaci o čistotě ovzduší v České republice. Tuto práci zakončuji popisem výsledné aplikace a zhodnocením samotné platformy Windows Runtime.

2 Windows Runtime

Se vstupem operačního systému Windows 8 na trh byl současně uveden také Windows Runtime. Windows Runtime neboli WinRT je moderní platforma společnosti Microsoft, které umožňuje vývojářům vytvářet takzvané Windows Store aplikace. Touto technologií vytvořené aplikace je možné spustit pouze na zařízeních s operačním systémem Windows 8.

Tato platforma nenahrazuje Win32 API, protože je v operačním systému Windows 8 plně podporováno. Přináší nový programovací model pro vytváření aplikací jak pro stolní a přenosné počítače založené na platformě x86, ale také pro přenosná zařízení, jejichž základ je postavený na ARM procesorech. Uživatelům a vývojářům tak přináší alternativu vůči ostatním mobilním platformám jako je Android nebo iOS.

2.1 Základní vlastnosti Windows Runtime

- Aplikace jsou spustitelné pouze na OS Windows 8.
- Podpora x86 a ARM platforem zároveň.
- Vývoj v jazycích, které již známe, tj. C++, C#, Visual Basic, JavaScript, HLSL.
- Grafické uživatelské rozhraní v XAML nebo HTML5 a CSS3.
- Aplikace beží v Sandboxu.
- Aplikace nemá přístup k citlivým zdrojům bez povolení, tj. adresáře, kamera, senzory, lokalizace apod.
- Distribuce aplikace je řešena přes Windows Store.

2.2 Programovací jazyky

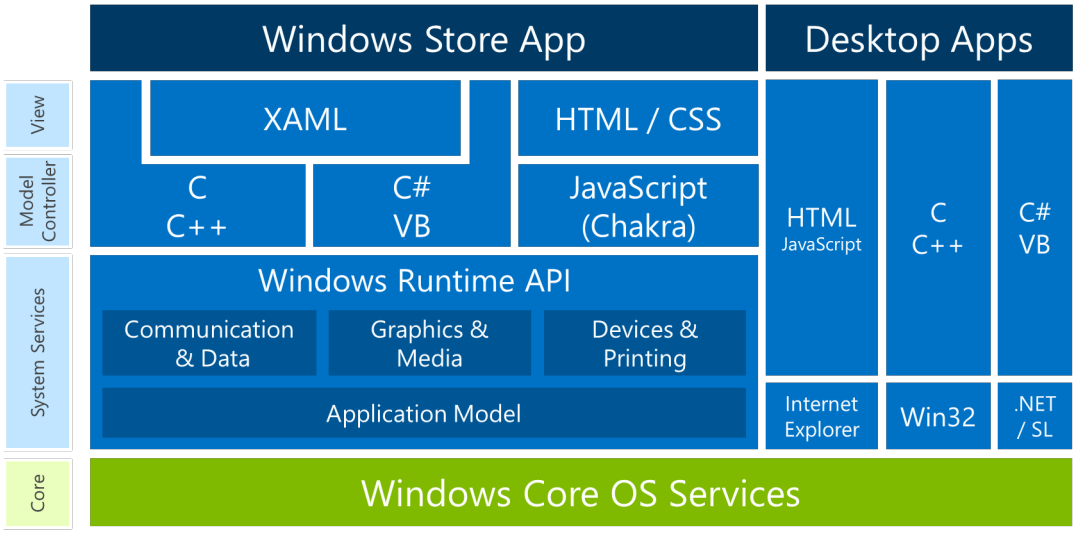
Mezi výhody vývoje aplikací patří možnost výběru programovacího jazyka díky jazykovým projekcím. Jazykové projekce mapují Windows Runtime API do několika programovacích jazyků, a tedy vývoj působí pro daný programovací jazyk přirozeně. K Windows Runtime tedy nepřistupujeme přímo ale pomocí jazykových projekcí.

K vývoji aplikační logiky si můžeme zvolit jazyky C++, C#, Visual basic nebo JavaScript.

Pro grafické uživatelské prostředí lze využít XAML nebo HTML5 a CSS3.

K dispozici je také DirectX a vývoj je umožněn skrze C++ a HLSL. XNA není pro vývoj aplikací podporováno.

Nejčastějšími kombinacemi aplikační logiky a UI je C++, C#, Visual basic a XAML nebo JavaScript a HTML5 s CSS3. Z výkonostních důvodů může být aplikační logika napsána C++, JavaScriptu a UI v HTML5 s CSS3. Vzájemná kombinace několika programovacích jazyků se tedy nijak nevylučuje.



Obrázek 1: Platforma Windows 8 [12]

2.3 Vývojové nástroje

Microsoft spolu s WinRT dodává vývojové nástroje.

- Visual Studio 2012
  - Visual Studio Express 2012 pro Windows (zdarma)
  - Visual Studio 2012 Ultimate
  - Visual Studio 2012 Premium
  - Visual Studio 2012 Professional
- Blend pro Visual Studio 2012
  - Návrh a vývoj UI
- .NET Framework 4.5
  - Zahrnuto ve Windows 8
  - Instalováno s Visual Studio 2012

2.4 Vývojářská licence

Vývojářská licence umožňuje vývojáři instalovat, vyvíjet a testovat Windows Store aplikace před tím než prochází procesem certifikace pro Windows Store a její publikací. Microsoft poskytuje licenci zdarma, a je ji třeba každých třicet dnů obnovovat. Vlastnictvím licence získáváme možnost spouštět necertifikované aplikace. Je také vázaná na konkrétní zařízení. Vývojářskou licenci a jeji obnovení je možné získat skrze Visual Studio 2012 nebo pomocí příkazové řádky a příkazem *Get-WindowsDeveloperLicense*.

3 Windows Store aplikace

Windows Store aplikace je novým typem aplikace, která je spustitelná na zařízeních s operačním systémem Windows 8. Typické pro ně je uživatelské rozhraní Windows 8 UI. Aplikace beží v pouze jednom okně, které je zobrazeno implicitně na celou obrazovku. Vzhledem k široké škále zařízení, na kterých jsou aplikace spustitelné, můžeme zadávat vstupy do zařízení ovládat několika způsoby jako dotek, stylus, klávesnice, myš atd. Aplikace jsou k tomuto ovládání uzpůsobeny. Místo klasických ikon zde nacházíme dlaždice, které mohou být v podobě takzvané živé dlaždice.

Instalace aplikace probíhá skrze Windows Store a odinstalace již na samotném zařízení. Instalované aplikace a jejich nastavení lze také synchronizovat mezi dalšími zařízeními s Windows 8.

3.1 Základní vlastnosti Windows Store aplikace

- Spustitelná na OS Windows 8
- Uživatelské rozhraní Windows 8
- Ovládání zařízení vícero způsoby, včetně dotyku, stylusu, myši a klávesnice
- Zobrazení v jednom okně implicitně na celou obrazovku
- Místo ikon existuji dlaždice, také v podobě živé dlaždice
- Sdílení obsahu mezi aplikacemi díky kontraktům
- Možnost synchronizace instalovaných aplikací a jejích nastavení

3.2 Uživatelské rozhraní

Jednou z nejzásadnějších a nejviditelnějších změn vůči standardním desktopovým aplikacím je uživatelské rozhraní Windows 8 UI. Koncept uživatelského rozhraní existuje již nějakou dobu, přestože je aktuálně aplikován v ekosystému Microsoftu mnohem více než kdykoliv jindy.

Využití principů Windows 8 UI si můžeme všimnout již u softwaru prvních Zune zařízení, jejichž menu bylo přehledné a soustředilo se na obsah. Přestože zařízení nebyly na trhu příliš úspěšné, principy UI byly přeneseny prvně do Windows Phone a vedly k vytvoření kompletně nového uživatelského rozhraní Windows 8 UI.

3.2.1 Úvodní Start nabídka

Úvodní Start nabídka je velkou změnou oproti předešlým verzím operačního systému Windows. Po přihlášení k uživatelskému účtu se defaultně zobrazí právě úvodní Start nabídka a ne pracovní plocha. V nabídce se místo ikon zobrazuji dlaždice v široké nebo úzké podobě. Desktopové aplikace mají vždy dlaždici úzkou s názvem aplikace v jejím zobrazení.

3.2.2 Charms

Charms je postranním panelem (menu) obsahující nabídku k hledání, sdílení, Start nabídku, položku zařízení a nastavení. Lze ji spustit najetím kurzoru do pravého horního rohu, klávesovou zkratkou Win + C nebo u dotykových zařízení táhnutím z pravé hrany obrazovky.

- **Panel hledání** - umožňuje ve Windows 8 a Windows RT nejen vyhledávat soubory a jejich obsah ale také možnosti nastavení, nainstalované aplikace včetně jejich obsahu.
- **Panel sdílení** - pomocí kontraktů mezi aplikacemi lze jejich obsah vzájemně sdílet.

3.2.3 AppBars

Windows Store aplikace často obsahují aplikační nabídku a to v podobně horní nebo spodní AppBar lišty.

Horní lišta je určena k navigaci mezi jednotlivými částmi aplikace. Spodní lišta je určena pro operace nad aplikací jako je například setřízení dat podle abecedy, export obsahu do PDF, uložení obrázku, přidání záznamu a podobně.

3.3 Životní cyklus aplikace

Klíčovým zdrojem každého zařízení, hlavně přenosného, je baterie. Správa Windows Store aplikací je v OS Windows 8 řešena jiným způsobem než u tradičních desktopových aplikací. Desktopovou aplikaci můžeme minimalizovat, maximalizovat a zavřít. Má neregulovaný přístup k procesoru, disku a síťovým zdrojům. U Windows Store aplikací se setkáváme se správou pomocí aplikačních stavů. Cílem stavů je co nejefektivnější správa zdrojů. Můžeme se setkat se čtyřmi aplikačními stavy: NotRunning, Running, Suspended a Terminated. Na obrázku 2 si můžeme všimnout jednotlivých stavů a jejich vzájemných přechodů.

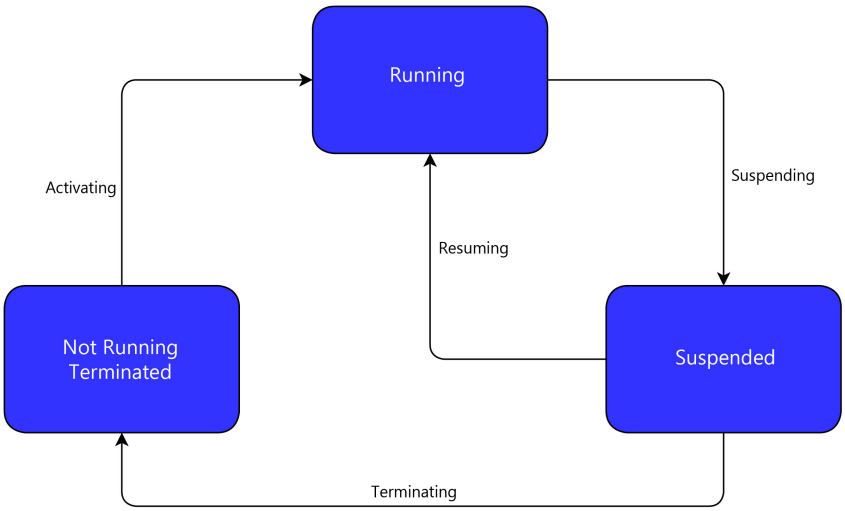
3.3.1 Not Running

Aplikace v tomto stavu není spuštěna a nevyužívá operační pamět, disk, síťové prostředky a ani jiné systémové zdroje.

3.3.2 Running

Pokud aplikaci spouštíme, potom přechází do stavu Running. Do tohoto stavu může přejít několika způsoby:

- Spuštěním aplikace pomocí dlaždice z nabídky Start.
- Otevřením typu souboru, který je asociován s aplikací.
- Využitím File Open Picker a File Save Picker.



Obrázek 2: Životní cyklus aplikace a přechody mezi aplikačními stavy

- Při možnosti vyhledávání v aplikaci.
- Je povolen příjem sdíleného obsahu.
- Běh aplikace na pozadí.
- Během funkce AutoPlay.

3.3.3 Suspended

Aplikační stav Suspended nastává jakmile aplikace neni na obrazovce zařízení viditelná. Do tohoto stavu přechází aplikace v reakci na událost OnSuspending, kdy má aplikace pět vteřin na uložení svého stavu, uživatelských dat a zastavení činnosti na pozadí.

Je doporučeno tyto data ukládat jednak v průběhu činnosti aplikace a také lokálně na uživatelově zařízení. Ve stavu Suspended zůstává celá aplikace v paměti, ale nevyužívá procesorového času, síťových zdrojů ani neoperuje nad diskem.

3.3.4 Resumed

Je-li aplikace ve stavu Suspended, může být obnovena, když se uživatel k aplikaci vrátí zpět nebo se Windows 8 vrací zpět z režimu kritického stavu baterie. Suspendovaná aplikace může pokračovat ve stavu, kde byla přerušena, protože je uchována v paměti. Pokud byla aplikace suspendována po delší dobu, tak je možné zaregistrováním události OnResuming potencialně zastaralá data aktualizovat.

3.3.5 Terminated

Aplikace při svém uzavírání přechází ze stavu Suspended do stavu Terminated. Přechod do stavu Terminated může nastat v případě zavření aplikace nebo z důvodu nedostatku paměti. Uživatel se o přechodu do tohoto stavu nedozví, protože v suspendované aplikaci již žádný kód neběží. Z hlediska významu je tento stav shodný s Not Running stavem.

4 Implementace Windows Store aplikace

4.1 Struktura aplikace

Každá Windows Store aplikace vytvořená v jazyce C# a XAML má svou specifickou strukturu, bez které by nebyla funkční. Každý projekt obsahuje několik souborů, jenž jsou pro každou aplikaci společné.

4.1.1 Package.appxmanifest

Tento XML soubor poskytuje Windows Runtime informace o Windows Store aplikaci. Přestože se jedná informace v XML podobě, lze jej jednoduše editovat v designeru vývojového prostředí Visual Studio 2012.

Dle zobrazení v designeru je možné informace rozdělit na části Application UI, Capabilities, Declarations a Packaging.

- **Application UI** - definuje balíček Windows Store aplikace a prvky ovlivňující vizuální podobu. Mimo jiné zde určujeme název, jazykovou verzi, popis, logo a splashscreen.
- **Capabilities** - jsou důležitou součástí, která definuje funkční schopnosti aplikace. Určuje ke kterým doplňkům nebo zařízením má aplikace přístup.

  V Package.appxmanifestu je nutné definovat tyto schopnosti. Bez jejich deklarace by nebylo možné doplňky nebo zařízení použít.

  Při prvním spuštění aplikace je ovšem uživatel vždy požádán o povolení k přístupu danému doplňku nebo zařízení. Lze zde využít například schopností geolokace, webové kamery, mikrofonu, přístup ke knihovnám jako je Hudba, Video, Obrázky.
- **Declarations** - v této části definujeme kontrakty pro ostatní aplikace, což vede k rozšíření funkcionality. Mezi kontrakty řadíme například Search Declaration, který umožní uživateli prohledávat aplikaci odkudkoliv z OS.
- **Packaging** - pro umístění aplikace na Windows Store je nutné balíček aplikace identifikovat. Údaje lze doplnit v této části Package.appxmanifest.

4.1.2 App.xaml

Tento soubor je vstupním bodem pro každou Windows Store aplikaci. Obsahuje globální správce událostí a inicializační kód pro datové modely.

4.1.3 Soubor pfx

Je dočasný soubor, který zajišťuje dočasnou certifikaci aplikace, aby mohla být spuštěna a testována na Windows 8 zařízeních.

4.2 XAML

Extensible Application Markup Language je značkový jazyk velice podobný XML k vytváření grafického rozhraní využívaných v technologiích jako je WPF, Silverlight, Windows Phone a Windows 8. Jednotlivé komponenty podobně jako v HTML a CSS mají v jazyce XAML svou roli. Řeší rozložení prvků v aplikaci, interakci s uživatelem či zobrazení obsahu a dat.

4.2.1 AppBar

V tradičních desktopových aplikací jsou možnosti menu řešeny přidáním ovládacího prvku Menu. Naprostá většina uživatelů je zvyklá na menu Soubor, Úpravy, Zobrazit a dále na horní části aplikačního okna. Z tohoto menu řídí činnost aplikace.

S Windows 8 přišla v tomto ohledu změna. Cílem Windows Store aplikace je dát přednost obsahu před prvky, které by mohly uživatelovu pozornost rozrušit. Menu a ikony nejsou zobrazeny. Je nutné tedy přemýšlet jiným způsobem. Namísto vytváření menu se stovkami možností dáme uživatelovi k dispozici operace, které jsou pro daná zobrazená data nejklíčovější. Musíme zvážit, jaké operace chce uživatel s daty provádět. Řešením je prvek AppBar.

AppBaru si můžeme všimnout již u Start nabídky po kliknutí na pravé tlačítko myši nebo po swipe gestu ze spodní hrany dotykového zařízení. Ve Windows Store aplikacích můžeme vytvořit spodní (Bottom AppBar) nebo horní (Top AppBar) nabídku. Horní nabídka slouží z návrhového hlediska pro navigaci mezi stránkami aplikace. Spodní nabídka obsahuje dostupné operace, které jsou nad zobrazenými daty dostupné.

```
<Page.BottomAppBar>
    <AppBar x:Name="bottomAppBar" Padding="10,0,10,0">
        <Grid>
            <StackPanel x:Name="rightPanel"
                    Orientation="Horizontal"  HorizontalAlignment="Right">
                <Button Style="{StaticResource AppBarButtonStyle}"
                    Content="&#xE174;"
                    AutomationProperties.Name="Sort"
                    AutomationProperties.AutomationId="SortButton" Tapped="
                        Button_Tapped_1" />
            </StackPanel>
        </Grid>
    </AppBar>
</Page.BottomAppBar>
```

Výpis 1: Bottom AppBar

4.2.2 ScrollViewer

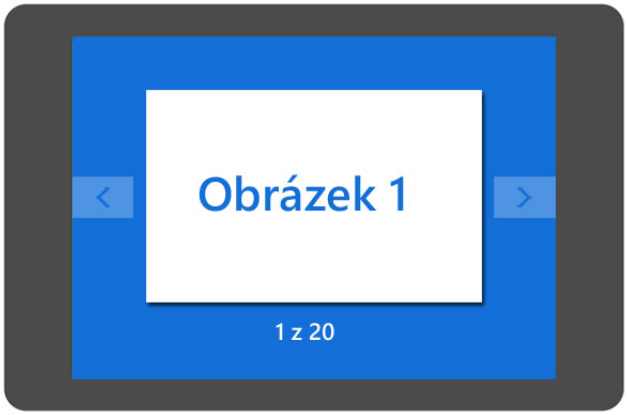
Prvek ScrollViewer řeší situaci, kdy zobrazený obsah přesahuje plochu obrazovky. Zároveň může sloužit jako alternativa k prvkům *Panorama* a *Pivot* u Windows Phone. Další obsah je možné zobrazit vertikálním a horizontálním posuvníkem. Posuvníky lze nastavit vlastnostmi *VerticalScrollBarVisibility* a *HorizontalScrollBarVisibility* na hodnotu Disabled

(nepovoleno), Enabled (povoleno) nebo Auto (automaticky dle rozsahu zobrazeného obsahu).

Obrázek 3: Zobrazení pomocí prvku ScrollViewer

4.2.3 FlipView

FlipView je ovládacím prvkem, který je navržen, aby zobrazil obsah kolekce po jednom prvku. Tento přístup zobrazení je velice praktický pro aplikace typu prohlížeč obrázků. Jedná se o velice efektní a designově dobře navrženou komponentu.



Obrázek 4: FlipView element pro zobrazení prvku z kolekce

4.2.4 GridView a ListView

TextBlock, ListBox, ComboBox a další podobné prvky jsou vhodné pro zobrazení jednoduchých dat a možností. Pro prezentování komplexnejších informací nejsou tyto prvky vhodné a v těchto případech využíváme GridView a ListView.

GridView je nejvyužívanějším ovládacím prvkem pro zobrazení dat ve Windows Store aplikacích. Zobrazení množiny dat nebo kolekce je typicky horizontální a zabírá větší množství pracovní plochy. GridView podporuje data binding včetně vytváření šablon (DataTemplate) pro kustomizované zobrazení dat. Datový zdroj předáváme pomocí vlastnosti *ItemsSource*. Vytvořením šablony *PanelItemsTemplate* můžeme také přepsat defaultní zobrazení dat v celém panelu.

Zobrazení dat pomocí ListView je mnohem bohatší než u ListBoxu či ComboBoxu a je typicky vertikální. Zabírá většinou menší část plochy oproti GridView. Stejně tak zde existuje podpora vytváření šablon pro kustomizované zobrazení dat. Předání dat do ListView je stejné jako u GridView.

4.2.5 Ostatní ovládací prvky

V XAML je k dispozici množství dalších ovládacích prvků grafického rozhraní. Seznam všech komponent najdeme na stránkách MSDN [4].

4.2.6 Callisto

Callisto je externí knihovna pro Windows Store aplikace založené na jazyku XAML. Autorem je Tim Heuer, Principal Program Manager Lead ve společnosti Microsoft, který má na starosti vývojářskou platformu a nástroje pro XAML ve Windows. Callisto poskytuje přidanou funkcionalitu pro XAML UI framework [7]. V současné době rozšiřuje XAML o tyto komponenty:

- **FlyOut** - plovoucí komponenta podobná prvku v HTML
- **Menu** - menu v nabídce AppBar
- **MenuItem** - položka v menu
- **SettingsFlyout** - vytvoření panelu pro položku Nastavení
- **Rating** - komponenta hodnocení pro Metro UI
- **LiveTile** - animované živé dlaždice pro aplikaci
- **Tilt** - efekt naklonění při pokliknání na okraje nebo rohy aplikace
- **BooleanToVisibilityConverter** - konvertor boolean na Visibility
- **LengthToBooleanConverter** - konvertor zkoumající délku řetězce na boolean
- **RelativeTimeConverter** - konvertor ukazující čas jako řetězec, například „Před hodinou“

4.3 Data binding

Data Binding je dalším důležitým konceptem při vývoji Windows Store aplikací. Smysl tohoto konceptu je zajištění toku dat do a z elementů grafického rozhraní. Jedná se o propojení datového zdroje a elemtu UI. Jednosměrný Data binding zajišťuje načtení dat do jednotlivých elementů UI, a jejich případnou aktualizaci při úpravě dat. Obousměrný Data binding rozšiřuje jednosměrnou komunikaci o aktualizaci datového zdroje, pokud byla data změna v UI. Datový zdroj předáváme do vlastnosti DataContext konkrétního UI elementu.

V následujícím kódu definujeme třidu Person, reprezentující data formuláře. Instance třídy Person se pomocí data bindingu a vlastnosti DataContext předá uživatelskému rozhraní.

```
public class Person
{
        public string  Name {get; set;}
        public string  Surname {get; set;}
        public string  Email {get;  set;}
}

Person newPerson = new Person();
newPerson.Name = "Jakub";
newPerson.Surname = "Dvorak";
newPerson.Email = "jakub.dvorak@email.cz";

this.DataContext = newPerson;
```

Výpis 2: Deklarace datové struktury a data binding

V následujícím XAML kódu bindujeme data do ovládacích prvků grafického rozhraní použitím klíčového slova Binding.

```
<Grid x:Name = "MyGrid">
    <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
    </Grid.RowDefinitions>

    <Grid.ColumnDefinitions>
            <ColumnDefinition />
    </Grid.ColumnDefinitions>

    <TextBlock x:Name="FirstName" Text="{Binding Name}" Foreground="White" Grid.Row="0"
        Grid.Column = "0" HorizontalAlignment="Center" />
    <TextBlock x:Name="Surname" Text="{Binding Surname}" Foreground="White" Grid.Row="1
        " Grid.Column = "0" HorizontalAlignment="Center" />
    <TextBlock x:Name="Email" Text="{Binding Email}" Foreground="White" Grid.Row="2" Grid.
        Column = "0" HorizontalAlignment="Center" />
</Grid>
```

Výpis 3: XAML a data binding

4.4 Asynchronní programování

Asynchronní programování je klíčovým prvkem Windows Runtime pro poskytnutí responsivního a rychlého uživatelského prostředí. Některé komponenty .NET Frameworku byly ve verzi 4.5 o asynchronní metody obohaceny. Přestože modely asynchronního programování již existují, dovedli vývojáři .NET Frameworku ulehčit psaní asynchronního kódu přidáním optimalizované knihovny *The Task Parallel Library* a klíčových slov *async* a *await*. Vývojář se soustředí tedy především na výstup kódu než na napsání asynchronní funkce.

Použitím klíčového slova *async* označíme metodu nebo lambda výraz pro kompiler jako asynchronní blok kódu. Metoda s tímto klíčovým slovem je určena jako asynchronní. Během svého vykonávání může dospět k instrukci, jejíž provedení trvá delší dobu. V tomto případě je nutné využít asynchronie, abychom předešli blokování aktuálního vlákna. Instrukce s klíčovým slovem *await* řekne kompileru, že daná asynchronní metoda nemůže pokračovat než je asynchronní proces ukončen. Protože asynchronní část funkce neblokuje aktuální vlákno, synchronní kód je exekuován. Návratový typ každé asynchronní metody může být *void*, *Task* nebo *Task<TResult>*.

4.5 Uložiště dat

WinRT disponuje jmenným prostorem *Windows.Storage*, který umožňuje operovat s datovými uložišti pro Windows Store aplikace. Tuto funkcionalitu využíváme, pokud chceme zapisovat nebo číst lokální nastavení aplikace, ukládat dat v souborech či přistupovat k místním knihovnám jako jsou složky Dokumenty, Obrázky, Video a další.

K dispozici jsou tři typy datových uložišť: lokální, roaming a dočasné. Jsou dostupné z vlastností *LocalFolder*, *RoamingFolder*, *TemporaryFolder* objektu *ApplicationData.Current*. Lokální uložiště se nachází na uživatelově zařízení. Roaming uložiště je vhodné využít, pokud chceme přenášet malé množství dat mezi zařízeními se stejným Microsoft účtem. Velikost dat je omezeno na 8 až 64 kilobyte. Data na dočasném uložišti jsou vymazány, pokud se nevyužívá. Instalační složku lze zpřístupnit skrze objekt *Package.Current.InstalledLocation*.

K uložišti lze také přistupovat přes *ms-appdata* protokol.

- **Pro lokální data:** ms-appdata:///local/
- **Pro roaming data:** ms-appdata:///roaming/
- **Pro dočasná data:** ms-appdata:///temporary/

4.5.1 Nastavení a aplikační data

Nastavení jsou aplikační data a objekty typu *ApplicationDataContainer*. Slouží k ukládání jednoduchých hodnot v aplikaci. Délka identifikátoru je omezena na 255 znaků a velikostně na 8 až 64 kilobyte. Nastavení lze také ukládat na uložiště typu roaming. Jedná se většinou o nastavení preferencí v aplikaci a stav aplikace. Není doporučeno takto ukládat data, která se často mění, protože tento postup vede ke snížení výkonu aplikace.

4.6 Geolokace a Bing Maps

Schopnost zjištění zeměpisné polohy uživatele, přesněji řečeno zařízení, je v posledních letech velice populární a často nedílnou součástí aplikací. Aplikace, jakou je například Foursquare, by se bez této schopnosti neobešla. Zjištění zeměpisné polohy je možné díky GPS senzoru. Moderní zařízení se ovšem neomezují pouze na GPS senzor, ale geopozici lze také zjistit z informací o připojení k síti nebo umístění BTS stanic.

K těmto účelům je ve Windows Runtime k dispozici třída *Geolocator*. V následující ukázce je prezentováno praktické použití této třídy. Vzhledem k náročnosti funkcí *GeolocatorStatusChanged* a *GeolocatorPositionChanged* je nutné jejich provádění asychronně. Možnost geolokace je nutné povolit jednak v nastavení projektu ve Visual Studiu, v záložce Capabilities a uživatel musí tuto schopnost povolit při spuštění aplikace.

```
public sealed partial class MainPage: Page
{
    private Geolocator locator;

    public MainPage()
    {
        this.InitializeComponent();
        this.locator = new Geolocator();
        this.locator.DesiredAccuracy = PositionAccuracy.Default;
        this.locator.StatusChanged += this.GeolocatorStatusChanged;
        this.locator.PositionChanged += this.GeolocatorPositionChanged;
    }

    private async void GeolocatorStatusChanged(Geolocator sender, StatusChangedEventArgs
        e)
    {
        await Dispatcher.RunAsync(CoreDispatcherPriority.Normal,() =>
        {
            //zde umistime kod pri zmene statusu lokalizace
            //status zjistime z e.Status
        });
    }

    private async void GeolocatorPositionChanged(Geolocator sender,
        PositionChangedEventArgs e)
    {
        await Dispatcher.RunAsync(CoreDispatcherPriority.Normal,() =>
        {
            //zde umistime kod pri zmene zemepisne polohy

            /*novou zemepisnou polohu ziskame z e.Position.Coordinate.Longitude
              a e.Position.Coordinate.Latitude
              */
        });

    }
}
```

Výpis 4: Geolokace pro Windows Store aplikace

4.6.1 Bing Maps

Bing Maps je online službou společnosti Microsoft poskytující uživatelům možnost hledat, zkoumat, plánovat a sdílet informace o specifických lokacích. Díky využití tradičních silničních map, označených leteckých pohledů a schopnosti hledání lokace je Bing Maps unikátní příležitostí pro aplikace využívající geolokaci [2].

Prekvizitami je mít nainstalováno *Bing Maps SDK* a vlastnit účet *Bing Maps Account*, kde vygenerujeme speciální přístupový klíč pro Windows Store aplikaci. Veškeré prerekvizity jsou zdarma. Máme-li k dispozici pozici zařízení, můžeme ji zobrazit na mapě. V následujícím výpise kódu vytvoříme v XAML komponentu *Map*, která reprezentuje zobrazení mapy v aplikaci. Hodnota atributu *Credentials* je vygenerovaný klíč pro přístup k Bing Maps v aplikaci.

```
<Maps:Map x:Name="MyMap" Credentials="
    AopdKOf1fz4wjp1ioVO0NUfju8PsQ0LrcOtg86YEW1iKR5gGWbkunVvrwjZD004z" Width="
    500" Height="500" />
```

Výpis 5: Komponenta Map v XAML

Kód funkce GeolocatorPositionChanged ve výpise 5 upravíme pro potřeby zobrazení v komponentě Map. Vytvoříme proměnnou location typu *Location* a do konstruktoru dosadíme získanou zeměpisnou šířku a délku. Funkcí *SetView* nastavíme zobrazení pozice na mapě a označíme polohu špendlíkem pomocí třídy *Pushpin*.

```
private async void GeolocatorPositionChanged(Geolocator sender, PositionChangedEventArgs e)
    {
        await Dispatcher.RunAsync(CoreDispatcherPriority.Normal,() =>
        {
            Location location = new Location(e.Position.Coordinate.Latitude, e.Position.
                Coordinate.Longitude);
            Puspin pin = new Pushpin();
            Maplayer.SetPosition(pin, location);

            this.MyMap.Children.Clear();
            this.MyMap.Children.Add(pin);
            this.MyMap.SetView(location);

        });

    }
```

Výpis 6: Příklad využití Bing Maps ve Windows Store aplikaci

Bing Maps SDK lze také využít pro zobrazení trasy mezi jednotlivými lokacemi, či zjištění aktuální dopravní situace, pokud jsou informace v dané lokalitě dostupné.

4.7 Globalizace aplikace

Windows i Windows Store jsou využívány milióny uživateli v různých zemích, lidmi mluvícími různými jazyky. Tento fakt umožňuje vývojářům cílit aplikaci na konkrétní trh nebo několik trhů současně. Vytvářet aplikaci pro každý trh zvlášť by bylo velice

časově náročné. Windows Runtime umožňuje globalizaci aplikace bez výrazných změn ve zdrojovém kódu.

Překlad uživatelského rozhraní je řešen vytvořením souboru typu *Resource File* pro každý jazyk. Tento soubor slouží jako slovník, je tedy ve formě hodnota a klíč. Klíč slouží jako identifikátor řetězce v UI, který chceme zobrazit v daném podporovaném jazyce. Slovníky jsou umístěny v projektu ve složkách pojmenovaných dle konvence BCP-47, například pro český jazyk existuje složka cs-CZ, pro anglický jazyk v USA složka en-US. Na následujícím obrázku je zobrazena možná podoba slovníku.

Řetězce ze slovníků načítáme do uživatelského rozhraní pomocí třídy *ResourceLoader*. V následujícím výpise do konstruktoru třídy *ResoureLoader* zadáme parametr rovnající se jménu slovníku, funkce *GetString* z této třídy zajistí získání řetězce dle zadaného klíče jako parametr. Získaný řetězec je dosazen do atributu Text komponenty *TextBlock* s označením txbMenuCurrent.

```
public sealed partial class MainPage: LayoutAwarePage
{
    private ResourceLoader loader;

    public MainPage()
    {
        this.InitializeComponent();

        loader = new ResourceLoader("AppResources");
        txbMenuCurrent.Text = loader.GetString("MenuCurrent").ToString();
    }
}
```

Výpis 7: Proces získání řetězce ze slovníku

5 Migrace Windows Phone aplikace

V rámci této práce vznikla ukázková aplikace, na které prezentuji základní vlastnosti Windows Store aplikací, ale také učiněné kroky pro migraci z platformy Windows Phone.

V rámci migrace řešíme v zasádě dva typy výzev. Musíme zajistit transfer uživatelského rozhraní do podoby využívající schopnosti Windows Store aplikací a upravit aplikační logiku aplikace, aby korespondovala z Windows Runtime API. XNA, jenž se využívá u aplikací s vyšším grafickým výkonem, není ve Windows Runtime nadále podporováno. Řešením je převést jej do DirectX použitím C++ a HLSL. Celková složitost migrace závisí na množství použitých APIs, které nejsou obsažena ve Windows Runtime.

5.1 Windows Phone

Windows Phone je relativně mladá mobilní platforma, která se na trhu objevila ke konci roku 2010. Microsoft upustil od platformy Windows Mobile, a vytvořil platformu naprosto novou s novým uživatelským prostředím Metro.

Microsoft Silverlight a XNA byly nabídnuty jako aplikační modely. Se vstupem Windows Phone 8 na trh se modely upravily na XAML a Direct3D. XAML stejně jako u Windows Store aplikací slouží pro vytvoření uživatelského rozhraní, C# a Visual basic tvoří logiku WP aplikace. Direct3D je určen pro aplikace potřebující schopnosti vyššího grafického výkonu, primárně je oblíben vývojáři her.

Každé Windows Phone zařízená musí také splňovat následující minimální hardwarové požadavky.

- Kapacitní displej se 4 a více kontaktními body
- Minimální rozlišení 480 x 800
- A-GPS, proximity senzor, akcelerometr, kompas, světlo
- Kamera s rozlišením 5 megapixelů a více, blesk a tlačítko pro spuštění kamery
- Multimédia
- 256 MB RAM a více, 8 GB flash a více
- GPU s podporou DirectX 9
- CPU s výkonem ARMv7 Cortex/Scorption a lepším

5.2 Migrace uživatelského rozhraní

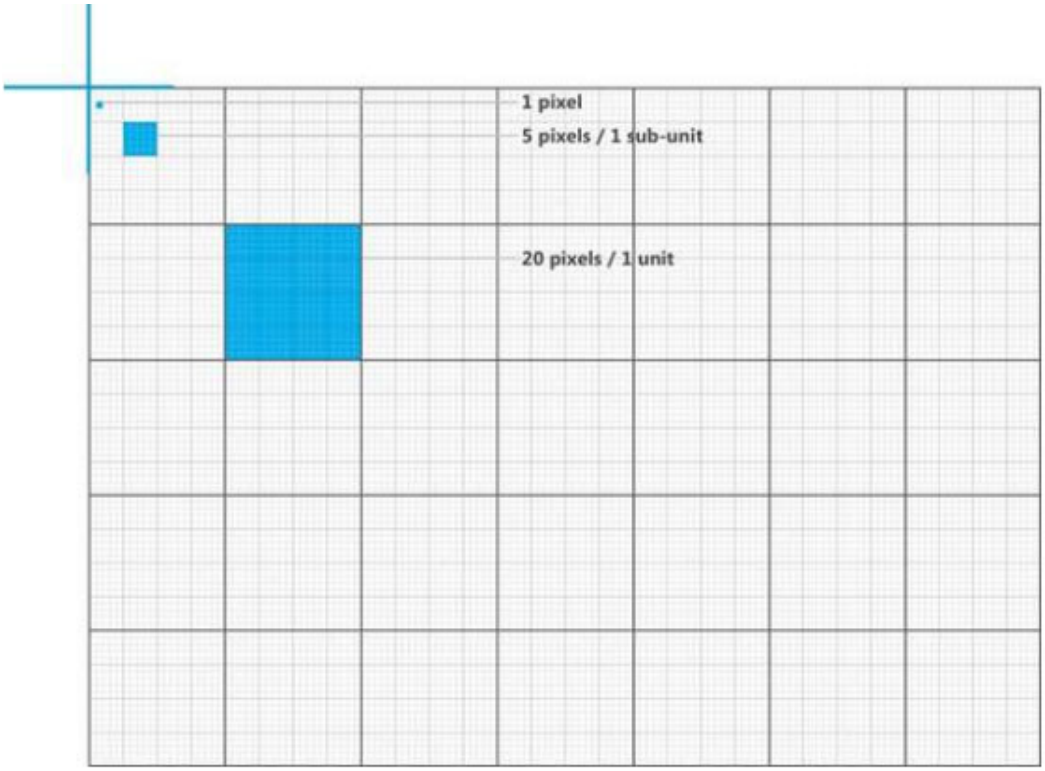
Uživatelské rozhraní aplikace je velice důležitým elementem každé aplikace. Ve Windows Store aplikacích to platí dvojnásob. Tvorba UI podléhá pravidlům, které tak zajišťují konzistentní chování každé aplikace umístěné na Windows Store. Při samotné migraci uživatelského rozhraní musíme zvážit několik věcí jako je například rozvržení obsahu (layout), navigaci v aplikaci, podporu *View States*, zahrnutí všech způsobů ovládání od dotyku po stylus, animace, logo a dlaždice a další.

5.2.1 Rozlišení obrazovky

Při migraci je důležité si uvědomit, že zařízení pro Windows Store aplikace mají jiné hardwarové vybavení než zařízení pro Windows Phone. Nejdůležitěším aspektem pro uživatelské rozhraní je rozlišení obrazovky. Standardní rozlišení obrazovky Windows Phone je 480 x 800 pixelů při orientaci na výšku, při orientaci na šířku 800 x 400 pixelů. Minimální podporované rozlišení zařízení pro Windows 8 je 1024 x 768 pixelů, optimální 1366 x 768 pixelů, kde je již podpora vizuálních stavů jako je například SnapedView. Maximální rozlišení není omezeno. To znamená, že máme větší prostor pro obsah. Musíme tedy zvážit, jestli nabídneme uživateli více obsahu nebo ne.

5.2.2 Systém mřížky

Pro rozložení obsahu ve Windows Store aplikacích využíváme systému mřížky. Tento system je zabudován do všech aplikací a výrazně ulehčuje samotný návrh aplikace a sjednocuje vzhled mezi jejími jednotlivými částmi. Mřížka je složena z jednotek a subjednotek. Základní měřítkem je jedna jednotka představující čtverec o velikosti 20 x 20 pixelů. Skládá se z 16 jednotek o velikosti 5 x 5 pixelů. Níže na obrázku 5 můžeme vidět grafické znázornění systému [10].



Obrázek 5: Systém mřížky pro návrh layout Windows Store aplikací

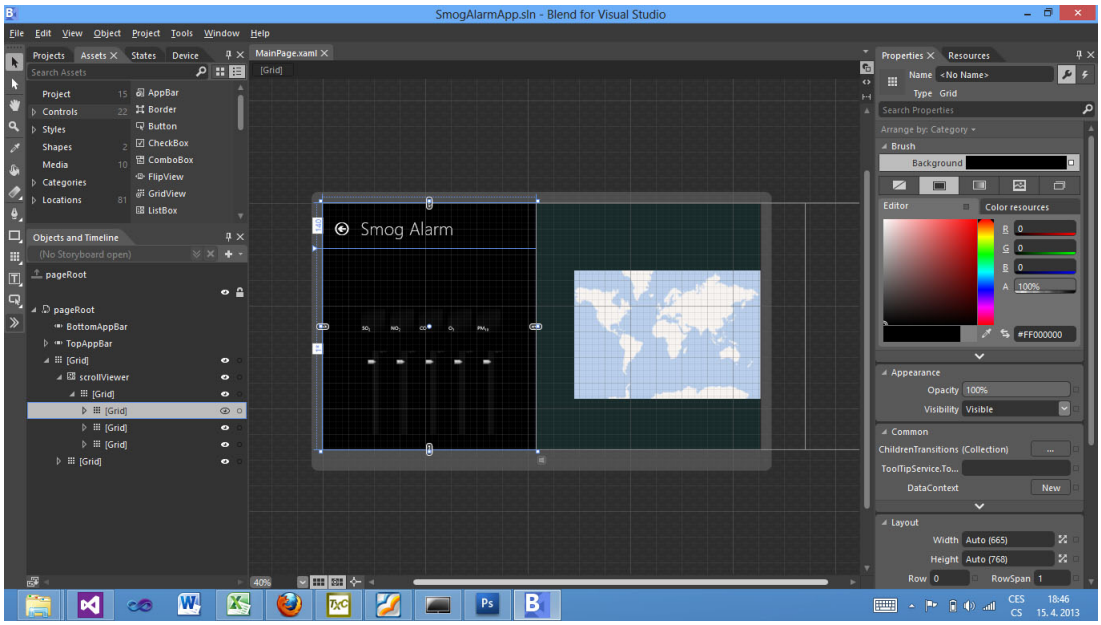
5.2.3 Fixní versus adaptivní layout

Pokud migrujeme herní aplikaci založenou především na bitmapových obrázcích, využíjeme s největší pravděpodobností metodu fixního layoutu. Zobrazení většího množství obsahu v těchto případech není možné nebo by nemělo výraznější smysl. Samotný návrh rozložení by měl začínat od rozlišení 1024 x 768 pixelů a 1366 x 768 pro podporu vizuálních stavů. Vhodné zobrazení UI elementů při vyšší hustotě pixelů (větším rozlišením) zajišťuje Windows Scaling. Windows Scaling zvětší element na 140 % nebo 180 % v závislosti na DPI. To znamená, že musíme grafické elementy dodat ve vektorovém formátu SVG, XAML, ve vysokém rozlišení v rastrovém formátu nebo poslední možností je dodání elementu ve třech rozlišeních odpovídající 100 %, 140 % a 180 %. Ze strany Microsoftu není tento typ layout ovšem příliš doporučován.

Pro ostatní typy aplikací je výrazně doporučeno využít adaptivního layoutu, který mění rozložení prvků na základě aktuální rozlišení obrazovky a zobrazí uživateli větší z pravidla množství informací. Obrazovka aplikace je nejčastěji rozdělena v mřížce na tři části: hlavička, navigační část a obsah. Zvolíme, které prvky layoutu budou fixní versus adaptivní v horizontálním a vertikálním směru. Zajistíme zobrazení všech kolekcí na 100 % délky i výšky obrazovky. k těmto účelům je vhodné využít XAML prvek ListView

nebo GridView. Pro zobrazení obrázků či mapy využijeme prvek Canvas, který automaticky vyplní volný prostor. Větší množství prostoru potom nejčastěji řešíme prostým zobrazením většího množství informací než na obrazovkách s nižším rozlišením.

Se samotným návrhem rozložení nemusíme začínat od počátku. K dispozici jsou předem definované šablony rozložení a formy navigace při zakládání projektů ve Visual Studio 2012. Více šablon bychom našli na stránkách projektu CodePlex [1]. Tyto šablony dodržují veškerá pravidla designu Windows Store aplikací. Svou aplikaci v různých rozlišení je možné testovat v simulátoru, který je také součástí IDE Visual Studio 2012. Chceme-li layout vytvořit od začátku využíváme nástroje Blend. Na obrázku 6 demonstruju návrh layoutu ukázkové aplikace v programu Blend.

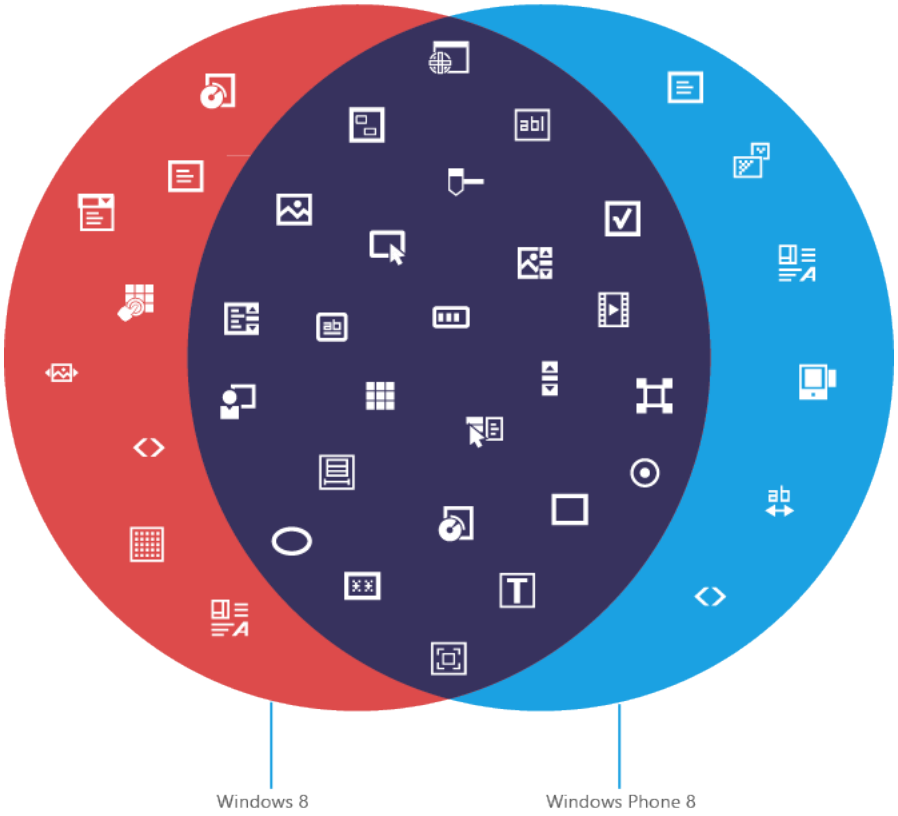


Obrázek 6: Návrh layoutu v programu Blend při využití systému mřížky.

5.2.4 XAML

Většinu XAML komponent jako je Grid, StackPanel, Canvas pro Windows Phone bychom nalezli také ve jmenných prostorech Windows Runtime [9]. Na obrázku 7 je grafické znázornění dostupných komponent pro jednotlivé platformy, což nám poskytne lepší obrázek o možnostech migrace.

Windows Runtime neobsahuje ovládací prvky rozložení jako jsou *Pivot* a *Panorama*, které jsou ve WP aplikacích velice populární. Tyto prvky jsou schopné zobrazit obsah, jenž svou velikostí přesahuje obrazovku zařízení. Ve Windows Runtime není možné Pivot a Panorama zcela funkčně nahradit, ale jako vhodnou alternativu se využívá FlipView, Grid nebo kombinace ScrollViewer a Grid, kterou jsem využil v případě migrace aplikace.



Obrázek 7: Grafické porovnání komponent XAML UI pro Windows Phone a Windows 8 [11]

```
<!--LayoutRoot is the root grid where all page content is placed.-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <!--Pivot Control-->
        <controls:Pivot  Title ="{Binding LocalizedResources.AppNameUC, Source={
             StaticResource Resources}}">
            <!--Pivot item one-->
            <controls:PivotItem Header="{Binding LocalizedResources.AppCurrent, Source={
                StaticResource Resources}}" DataContext="{Binding CurrentStation}">
                ....
            </controls:PivotItem>

            <!--Pivot item two-->
            <controls:PivotItem Header="{Binding LocalizedResources.AppMap, Source={
                StaticResource Resources}}">
                .....
```

```
            </controls:PivotItem>
                .....
        </controls:Pivot>
    </Grid>
```

Výpis 8: Pivot komponenta ve Windows Phone

```
<Grid Style="{StaticResource␣LayoutRootStyle}" RenderTransformOrigin="0.559,0.448">
        <Grid.RowDefinitions>
            <RowDefinition Height="140"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

            <ScrollViewer x:Name="scrollViewer" Grid.RowSpan="2" Style="{StaticResource␣
                HorizontalScrollViewerStyle}">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="665"/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                  </Grid>
                  <!-- Grid first part - Current situation -->
                  <Grid>
                            <Grid.RowDefinitions>
                        <RowDefinition Height="140" />
                        <RowDefinition Height="*" />
                    </Grid.RowDefinitions>
                            <Grid x:Name="ContentCurrent" Grid.Row="1" Margin="120,0,␣0,␣0"
                                >
                            </Grid>
                  </Grid>

                   <!-- Grid second part - Map -->
                  <Grid Grid.Column="1">
                            <Grid.RowDefinitions>
                        <RowDefinition Height="140" />
                        <RowDefinition Height="*" />
                    </Grid.RowDefinitions>
                            <Grid x:Name="ContentCurrent" Grid.Row="1" Margin="120,0,␣0,␣0"
                                >
                            </Grid>
                  </Grid>

                  <!-- Grid third part -Historical Data -->
                  <Grid Grid.Column="2">
                            <Grid.RowDefinitions>
                        <RowDefinition Height="140" />
                        <RowDefinition Height="*" />
                    </Grid.RowDefinitions>
                            <Grid x:Name="ContentCurrent" Grid.Row="1" Margin="120,0,␣0,␣0"
                                >
                            </Grid>
                  </Grid>
```

```
</Grid>
```

Výpis 9: Nahrazení Pivot komponenty za kombinaci ScrollViewer a Grid

5.3 Migrace aplikační logiky

Původní Windows Phone aplikace je napsána v jazyce C#. Díky jazykovým projekcím lze možné využít C# i v případě vývoj Windows Store aplikace. Tato podpora umožňuje znovuvyužití značné části původního kódu. Většina funkcionality známé v Silverlight a WP aplikacích najdeme i ve Windows Runtime.

5.3.1 Obecně

Postup migrace aplikační logiky Windows Phone aplikace lze provést v následujících krocích:

- Vytvoříme projekt z předem definované šablony pro Windows Store aplikace.
- Náhradíme původní jmenné prostory.
- Výkonově náročné metody přepíšeme do asynchronní podoby.
- Zkopírujeme ostatní zdrojový kód.
- Zabudujeme specifika Windows Store aplikací.
- Otestujeme aplikace nástrojem Windows App Certification Kit.
- Aplikaci nasadíme na Windows Store.

5.3.2 Jmenné prostory

Funkcionalita u Silverlight a WP aplikacích se nachází ve jmenných prostorech *System*, *Microsoft* a *Microsoft.Phone.Controls*. Pro Windows Runtime se nacházejí ve jmennných prostorech *Windows*. V tabulce 1 jsou zobrazeny zásadnější změny ve jmenných prostorech pro WinRT. V kódu je tedy zaměnit potřebné jmenné prostory.

5.3.3 Získání dat

Zobrazované informace v aplikaci musíme kvůli aktualizacím hodnot získávát z externího zdroje. Ve Windows Phone aplikaci pro stahování dat využíváme třídy *WebClient*, která ve Windows Runtime nadále neexistuje. Jako vhodnou alternativu se využívá třída *HttpClient*, která je schopna zprostředkovat komunikace mezi aplikací a externím zdrojem dat. V následujícím výpisu 10 je zobrazena metoda *DownloadData* přijímající URL adresu jako jediný parametr. Na dané URL poskytuje externí zdroj data v podobě řetězce, který se v aplikaci posléze parsuje.

```
private async Task<string> DownloadData(Uri uri)
{
    string content;
    try
    {
        HttpClient client = new HttpClient();
        HttpResponseMessage response = await client.GetAsync(uri);

        response.EnsureSuccessStatusCode();
        content = await response.Content.ReadAsStringAsync();

    }
    catch (HttpRequestException e)
    {
        content = "";
    }

    return content;

}
```

Výpis 10: Metoda DownloadData pro získání dat z externího zdroje

5.3.4 Ukládání a čtení dat z datového uložiště

V případě, že nelze získat data z externího zdroje například z důvodu nedostupného internetového připojení, aplikace využije informací z předešlé seance. Data jsou ukládána lokálně na uživatelův účet v podobě textového souboru. V následujících dvou výpisech je ilustrován způsob zápisu a čtení dat z lokálního uložiště v metodach *WriteRawData* a *ReadRawData*.

```
private async Task WriteRawData(string filename, string data)
{
    // settings options
    var option = Windows.Storage.CreationCollisionOption.ReplaceExisting;

    try
    {
        var file = await defaultFolder.CreateFileAsync(filename, option);
        MemoryStream saveData = new MemoryStream();
        XmlSerializer x = new XmlSerializer(data.GetType());
        x.Serialize(saveData, data);

        if (saveData.Length > 0)
        {
            using (var fileStream = await file.OpenStreamForWriteAsync())
            {
                saveData.Seek(0, SeekOrigin.Begin);
                await saveData.CopyToAsync(fileStream);
                await fileStream.FlushAsync();
            }
```

```
                return;
            }

        }
        catch { }

    }
```

Výpis 11: Ukládání dat do lokálního uložiště na uživatelově účtu.

```
private async Task<object> ReadRawData(string filename, System.Type type)
{
    try
    {
        var file = await defaultFolder.GetFileAsync(filename);
        using (IInputStream inStream = await file.OpenSequentialReadAsync())
        {
            XmlSerializer x = new XmlSerializer(type);
            return x.Deserialize(inStream.AsStreamForRead()).ToString();
        }
    }
    catch (FileNotFoundException) { return null; }
}
```

Výpis 12: Čtení dat ze souboru uloženém na lokálním uložišti

5.3.5 Geolokace

Aby aplikace mohla zobrazit informace o stavu ovzduší z nejbližší měřící stanice, je nutné určit aktuální polohu zařízení. Ve výpisu 13 je zobrazena metoda *StartGeolocation*, která vrací aktuální polohu zařízení. Původní metoda byla přepsána do asynchronní podoby a využívá nově třídu Geolocator od původní *GeoCoordinateWatcher* ve Windows Phone aplikaci. Vlastností *DesiredAccuracy* nastavujeme požadovanou přesnost zjištěné polohy. Je možné nastavit hodnotu Default nebo High. Hodnota Default optimalizuje pro delší výdrž baterie a výkon. Hodnota High zajišťuje co nejpřesnější hodnoty polohy, přičemž může využít i placených služeb, je náročnější z hlediska napájení a může degradovat výkon systému. Zavoláním funkce *GetGeopositionAsync* započne samotné zjišťování polohy. Metoda navrací hodnotu přiřazenou do proměnné myPosition typu *Geoposition*, původně byla proměnná typu *GeoCoordinate*, jenž ve WinRT v této podobě neexistuje.

```
private async Task<Geoposition> StartGeolocation()
{
    Geoposition myPosition;

    this.locator = new Geolocator();
    this.locator.DesiredAccuracy = PositionAccuracy.High;
    myPosition = await this.locator.GetGeopositionAsync();

    return myPosition;
}
```

Výpis 13: Určení polohy zařízení

5.3.6 Callisto a menu Charms

Knihovna Callisto je v aplikaci využita pro vytvoření nabídky v menu Charms. Z knihovny využívám komponentu Flyout, která není v XAML namespace dostupná. V metodě MainPage_CommandsRequested vytváříme pomocí třídy SettingsCommand jednotlivé položky v menu Charms. Konstruktor přijímá jako své parametry jedinéčné ID položky v menu, jméno položky pro zobrazení a zpracování při zvolení položky z menu. Při zvolení položky about je pomocí Callisto vytvořen Flyout a do něj načten obsah ze souboru. Samotný kód pro vytvoření položek je ve výpise 14.

```
void MainPage_CommandsRequested(SettingsPane sender,
    SettingsPaneCommandsRequestedEventArgs args)
{
    SettingsCommand privacy = new SettingsCommand("privacy", loader.GetString("
        SecurityAppHeader"), async (uicommand) =>
    {
        await Windows.System.Launcher.LaunchUriAsync(new Uri("http://url_addresss/
            privacypolicy.html"));
    });

    SettingsCommand about = new SettingsCommand("About", loader.GetString("
        AboutAppHeader"), e =>
    {
        SettingsFlyout flyout = new SettingsFlyout();
        flyout.FlyoutWidth = SettingsFlyout.SettingsFlyoutWidth.Wide;
        flyout.HeaderText = loader.GetString("AboutAppHeader");
        flyout.Background = new SolidColorBrush(Colors.White);
        flyout.HeaderBrush = new SolidColorBrush(Color.FromArgb(255, 0, 140, 0));
        flyout.Content = new About();
        flyout.IsOpen = true;
    });

    args.Request.ApplicationCommands.Add(privacy);
    args.Request.ApplicationCommands.Add(about);
}
```

Výpis 14: Flyout pomocí Callisto

5.4 Výsledná aplikace

Na obrázku 8 a 9 je zobrazena původní a migrovaná aplikace ve finální podobě. Při implementaci jsem se snažil co nejvíce zachovat grafickou a funkční podobu původní Windows Phone aplikace. Výsledná aplikace je schopna získávat data o stavu ovzduší z externího zdroje, zjistit aktuální pozici zařízení, tyto informace zpracovat a zobrazit v přehledné podobě.

Pro aplikaci bylo nutné navrhnout nové uživatelské prostředí v XAML, jenž by bylo podporováno ze strany Windows Runtime. Využil jsem prvku ScrollViewer a systému prvků typu Grid, abych se přiblížil původně použitému prvku Panorama. Dále bylo nutné zakomponovat ovládání aplikace, což jsem vyřešil dle standardu zabudováním Top AppBar nabídce. Využil jsem také komponent třetích stran, konkrétně knihovnu *Callisto*, kde jsem využil komponenty *FlyOut*.

U aplikační logiky bylo nutné změnit jmenné prostory a nahradit funkčnost některých tříd alternativou. Příkladem může být třída *HttpClient*. Hlavní změny aplikační logiky byly provedeny u náročnějších metod, které mohly potenciálně zpomalovat plynulost výsledné aplikace. Tyto metody jsem převedl do asynchronní podoby. Pro účely zobrazení hodnot kvality ovzduší na mapě jsem využil služeb Bing. Aplikace je také lokalizována do dvou jazyků, což bylo možné díky načítání hodnot z připravených slovníků. Je také schopná ukládat data pro zobrazení na bezpečné lokální uložiště pro případ nedostupnosti senzorů nebo internetového připojení. Tato funkce je dostupná po prvním spuštění aplikace, kdy jsme schopni získat data alespoň jednou.

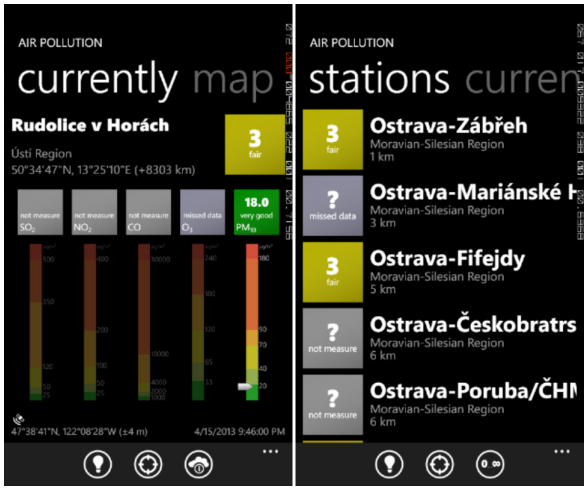
Hlavní obrazovka aplikace je rozdělena na tři části: Aktuální stav (Current Situation), Mapa (Map) a Historická data (Historical data). Na hlavní obrazovce a stránce Měřící stanice (Stations) jsme schopni vyvolat Top AppBar nabídku obsahující položky Home, Stations a Locate. Zároveň vyvoláním menu Charms objevíme, části Settings (Nastavení), položky O aplikaci (About app) a Ochrana osobních údajů (Personal Security).

V první části je zobrazen název měřící stanice a celkový stav ovzduší v dané lokalitě hodnocený známkou. Defaultně je zobrazena nejbližší měřící stanice s dostupnými daty vzhledem k poloze zařízení. Dále jsou zobrazeny hodnoty a grafy stavu jednotlivých chemických prvků a polétavého prachu.

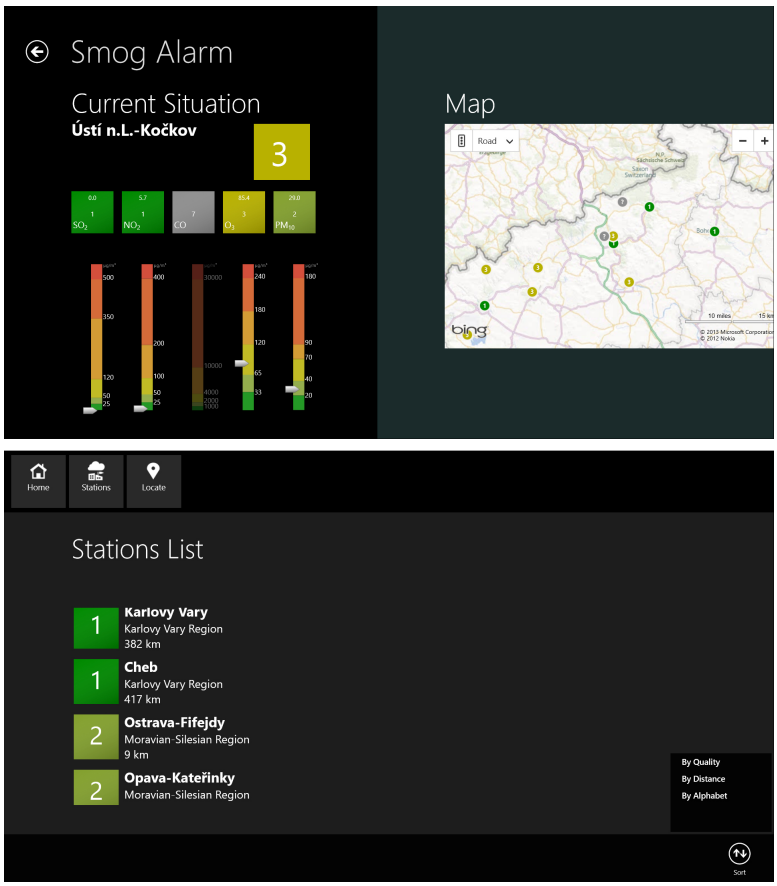
Ve druhé části aplikace zobrazuje mapu se svou aktuální polohou (světle zelený špendlík) a polohu měřících stanic v okolí zařízení. Stanice jsou označeny špendlíky s naměřenou hodnotou celkového stavu ovzduší. Na mapě je možné se pohybovat, přibližovat si a měnit pohled například na pohled letecký. Při pohybu jsou zobrazeny všechny měřící stanice na území České republiky.

Ve třetí, poslední části je zobrazena statistika celkové kvality ovzduší za posledních 30 dnů dle známky a grafy zobrazující historická data.

Uživatel má možnost si aktuální měřící stanici změnit zvolením jiné na stránce Měřící stanice (Stations), která obsahuje seznam stanic s aktuálním hodnocením, regionem, kde se nachází a vzdáleností od zařízení, pokud je informace dostupná. Stanice lze seřadit dle vzdálenosti od zařízení, aktuálního stavu ovzduší či podle abecedy. Zvolením nové stanice je uživatel navigován zpět na hlavní obrazovku, kde jsou zobrazena data zvolené stanice.



Obrázek 8: Původní Windows Phone aplikace



Obrázek 9: Výsledná Windows Store aplikace

6 Zhodnocení platformy Windows Runtime

Windows Runtime je velice mladou platformou, a proto její integrace a akceptace ve světě informačních technologií jistou dobu ještě potrvá. Nicméně společnost Microsoft v současné době již pracuje na nové verzi OS Windows s kódovým označením Blue, což svědčí o důvěře společnosti v tuto platformu.

Jednou z hlavní výhod WinRT je využitelnost na širokém rozpětí zařízení. Se shodnou Windows Store aplikací se můžeme setkat na stolním počítači, notebooku nebo tabletu s ARM procesorem. Může se jednat o zařízení s dotykovým displejem i bez něj. Tento fakt usnaďnuje vývojářům práci, protože není nutné škálovat aplikaci pro různé typy zařízení. Proces vývoje je také rychlejší v porovnání s vývojem aplikace pro platformu x86 a libovolnou mobilní platformu zvlášť.

Již jsem v práci zmiňoval, že Windows Store aplikace jsou typické svým uživatelským rozhraním, označovaném jako Windows 8 UI. Využitelnost rozhraní je dle mého mnohem větší na zařízeních s dotykovým displejem, kde práce se systémem i aplikacemi je svižnější, více intuitivní oproti ovládání myší, klávesnicí či jiným vstupním zařízením. Můj názor podporuje i aktuální vzestup popularity zařízení s dotykovou obrazovkou. Nabídka produktů na českém území se od vstupu Windows 8 na trh rozšířila. Také sama společnost Microsoft vyvinula tablet Surface. Z pohledu vývoje má rozhraní přinos již ve svém samotném návrhu, kdy obsah má přednost před spoustou nabídek a možnostmi nastavení. Přináší také živé dlaždice, které u desktopových aplikací nenajdeme. Existuje zde také možnost zabudování ovládání aplikace gesty.

Vývoj samotného uživatelského rozhraní v XAML či HTML5 a CSS3 je složitější zejména v návrhu, který podléha standardům. V tomto ohledu je vhodné využít služeb designéra či zkušeného vývojáře.

Další výhodou WinRT je dostupný výběr programovacích jazyků. Vývojáři, kteří jsou zvyklí na C# a .NET, většinu svých znalostí mohou využít a neučí se technologii od začátku. Složitost pro webové vývojáře s HTML5 a JavaScriptem není také příliš velká. Jistou nevýhodou může být vývoj her, pokud jsme zvyklí na XNA framework. Zde je opravdu nutné přistoupit k C++ a DirectX. Zajímavé je kombinování technologií pro uživatelské rozhraní a aplikační logiku, ať už z výkonostních či jiných důvodů.

Mezi další pozitiva platformy považuji synchronizaci nastavení a souborů pomocí SkyDrive, sdílení obsahu mezi aplikacemi skrze kontrakty.

Otázka pro mnohé vývojáře je, zdali zvolit desktopovou aplikaci nebo Windows Store aplikaci. Oba typy mají své výhody a jsou vhodné pro různé účely, a proto uvedu nejdříve srovnání.

6.0.1 Windows Store aplikace

- Distribuce a monetizace pouze přes Windows Store.
- Aplikace mohou komunikovat mezi sebou pomocí menu Charms.
- Aplikace jsou spustitelné pouze na Windows 8.

6.0.2 Desktopová aplikace

- Aplikace nejsou spustitelné na ARM procesorech.
- Aplikace jsou zpětně kompatibilní s Windows 7.
- Vyspělá vývojová platforma.

Desktopové aplikace mají výhodu ve své robustnosti, plném přístupu k operačnímu systému a důvěryhodnosti u uživatelů. Vzhledem k těmto vlastnostem jsou vhodnější pro podnikové aplikace. Je zde také větší podpora APIs třetích stran.

Windows Runtime je stále vyvíjející se platforma. Microsoft nabídnul pro vývojáře relativně snadný přesun z desktopových a webových aplikací, což otevřelo dveře na nový trh a nové příležitosti.

7 Závěr

V této bakalářské práci jsem měl za cíl popsat platformu Windows Runtime, navrhnout a implementovat ukázkovou aplikaci, která by využívala specifika aplikací pro Windows 8. Jednotlivé kapitoly jsou řazené od obecných informací po konkrétnější, aby čtenář mohl získat solidní základnu pro vývoj aplikací. Ukázal jsem možnosti implementace a také migrace aplikace z platformy Windows Phone. Výsledkem práce je tedy jednak průvodce pro vývojáře ale také zmigrovaná aplikace využívající vlastnosti Windows Store aplikací.

V první kapitole jsem se věnoval představení platformy Windows Runtime, jaké jsou základní rysy a rozdíly od Win32 aplikací. Stejně tak jsem poskytl informace o nástrojích, které je nutné mít, abychom se samotným vývojem mohli začít. V dalších dvou kapitolách popisuju vlastnosti a způsob implementace Windows Store aplikace. Vybral jsem dle mého názoru důležité témata, které jsou pro aplikace charakteristické a při jejich vývoji se s nimi s největší pravděpodobností setkáme. Zahrnul jsem také možnosti rozšíření o služby Bing a knihovny Callisto.

Poslední kapitola se věnuje ukázkové aplikaci, která zároveň ukazuje vhodný způsob migrace z platformy Windows Phone. Aplikaci jsem testoval na stolním počítači a v simulátoru. Z pohledu dalšího vývoje aplikace by bylo ideální využít služeb designéra a navrhnout nové uživatelské rozhraní. Funkcionalita by byla jednoduše rozšířitelná přidáním komponent z nástrojů společnosti Telerik nebo přidáním dalších funkcí pro Bing Maps. Vhodná by také byla výkonová optimalizace.

Zadání této bakalářské práce byla pro mě výzvou, protože jsem neměl velké zkušenosti z platformou Windows Runtime ani Windows Phone. Práce byla pro mě velkým přínosem. Podíval jsem se pod pokličku dvou platforem společnosti Microsoft a naučil jsem se vyvíjet aplikace pro WinRT platformu. Tyto znalosti v budoucnu jistě zužitkuji, protože mě oblast vývoje moderních aplikací velice zajímá.

8 Reference

[1] BECHYNSKÝ, Štěpán. *Microsoft Techdays (přednáška)*, Ostrava, 6.11.2012

[2] *Bing Maps, Microsoft, MSDN Library* [online], [cit. 10.4.2013]. Dostupné z WWW: http://msdn.microsoft.com/en-us/library/dd877180.aspx

[3] BROCKSCHMIDT, Kraig. *Programming Windows: 8 Apps with HTML, CSS, and JavaScript*. Microsoft Press, 2011, ISBN 9780735672611.

[4] *Controls list (Windows Store apps using C/VB/C++ and XAML), Microsoft, MSDN library* [online], Dostupné z WWW: http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh465351.aspx

[5] FREEMAN, Adam. *Metro revealed: Building Windows 8 apps with XAML and C#*. New York, Apress, 2012, ISBN 1430244887.

[6] FREEMAN, Adam. *Metro revealed: Building Windows 8 apps with HTML5 and Javascript*, New York, Apress, 2012, ISBN 978-1430244882.

[7] HEUER, Tim. *Callisto* [online], Dostupné z WWW: https://github.com/timheuer/callisto

[8] MAMTA DALAL, Ashish Ghoda. *XAML developer reference*. Sebastopol, Calif: O'reilly Media. ISBN 978-073-5658-967.

[9] *Migrating a Windows Phone 7 app to a Windows Store app using XAML, Microsoft, MSDN library* [online], Dostupné z WWW: http://msdn.microsoft.com/en-us/library/windows/apps/hh465136.aspx

[10] *Windows 8 User Experience Guidelines, Microsoft* [online], Dostupné z WWW: http://www.microsoft.com/en-us/download/details.aspx?id=30704

[11] *XAML controls comparison between Windows Phone 8 and Windows 8, Microsoft, MSDN library* [online], Dostupný z WWW: http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj735581(v=vs.105).aspx

[12] *Začínáme psát aplikace pro Windows Store v HTML5* [online]. [cit. 10.4.2013]. Dostupné z: http://www.zdrojak.cz/clanky/zaciname-psat-aplikace-pro-windows-store-v-html5/

A CD

Na přiloženém CD je obsaženo:

1. Bakalářská práce
2. Zdrojový kód Windows Store aplikace
3. Testovací soubory obsahující nerozparsovaná zobrazovaná data
4. Video zachycující činnost výsledné aplikace

B Srovnání jmenných prostorů

API	WP API	WinRT API
XAML	System.Windows.*	Windows.UI.Xaml.*
Devices	Microsoft.Devices, Microsoft.Devices.Radio, Microsoft.Devices.Sensors	Windows.Devices.Enumeration, Windows.Devices.Enumeration, Windows.Devices.Enumeration.Pnp, Windows.Devices.Input, Windows.Devices.Sensors
Maps	Microsoft.Phone.Info, Microsoft.Phone.Notification, Microsoft.Phone.Reactive, Microsoft.Phone.Shell, Microsoft.Phone.Tasks	Windows.Devices.Sms, Windows.ApplicationModel.Background, Windows.ApplicationModel.Contacts, Windows.ApplicationModel.Core
Maps	Microsoft.Phone.Controls.Maps.*	Windows.Devices.Geolocation
Marketplace	Microsoft.Phone.Marketplace	Windows.ApplicationModel.Store
Networking and syndication	Microsoft.Phone.Net, Microsoft.Phone.Net.NetworkInformation	Windows.Networking, Windows.Networking.BackgroundTransfer, Windows.Networking.Connectivity, Windows.Networking.NetworkOperators, Windows.Networking.Sockets, Windows.Web.AtomPub, Windows.Web.Syndication
Programming and data models	System	Windows.Foundation, Windows.Foundation.Collections, Windows.Foundation.Metadata, Windows.Data.Xml.Dom, Windows.Data.Xml.Xsl Windows.Data.Json
Location	System.Device.Location	Windows.Devices.Geolocation, Windows.Networking.Proximity
Automation and diagnostics	System.Diagnostics, System.Windows.Automation.Peers	Windows.Foundation.Diagnostics
Graphics (3D transforms)	XNA Framework Class Library, Content Pipeline Class Library	Neexistuje ekvivalent. Lze použít DirectX.
Controls and UI infrastructure	Microsoft.Phone.Controls, Microsoft.Phone.Controls.Primitives	Windows.UI.ApplicationSettings, Windows.UI.Core Windows.UI.Input, Windows.UI.Notifications, Windows.UI.ViewManagement
Storage	System.IO.IsolatedStorage class	Windows.Storage, Windows.Storage.FileProperties

Tabulka 1: Jmenné prostory Silverlight/WP versus WinRT